REMARKS

This amendment is responsive to the Office Action dated February 11, 2008. Applicant has amended claims 1, 18, 35 and 47 and added new claims 65–68. Claims 1–8, 18–25, 35–41, 47–54 and 65–69 are pending.

Claim Rejection Under 35 U.S.C. § 103

In the Office Action, the Examiner rejected claims 1–5, 7–8, 18–22, 24, 25, 40, 47–51, and 53–54 under 35 U.S.C. 103(a) as being unpatentable over Drews (US 6,463,535) in view of IEEE Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practice (hereafter "IEEE") and Kozen "Efficient Code Certification". Applicant respectfully traverses the rejection to the extent such rejections may be considered applicable to the claims as amended. The applied references fail to disclose or suggest the inventions defined by Applicant's claims, and provide no teaching that would have suggested the desirability of modification to arrive at the claimed invention.

With reference to independent claim 1, for example, the applied references lack any teaching that would have suggested a method comprising, upon power-up of a computer, retrieving boot code and a certificate from a peripheral device coupled to the computer, the certificate describing operation of the boot code for initializing the peripheral device, wherein the boot code is generated from a first programming language, and wherein the certificate includes an annotation defining a proof of security and safety for both (i) one or more blocks of code generated from a second programming language different from the first programming language and (ii) one or more corresponding blocks of the boot code resulting from translation of the one or more blocks of the code of the second programming language into the first programming language, as required by Applicant's currently amended claim 1.

The applied references also lack any teaching that would have suggested the method comprising verifying, with the computer, security of the boot code associated with the peripheral device by performing a security check on the boot code in accordance with the certificate, and executing the boot code with the computer to (i) initialize the peripheral device based on a result of the security check and (ii) provide, subsequent to the initialization, an interface by which the

computer controls operation of the peripheral device, as further required by Applicant's currently amended claim 1.

Applicant notes first that the Kozen reference lacks any teaching to suggest application of the techniques described in Kozen to boot code. Moreover, Applicant submits that the Examiner is improperly using hindsight to piecemeal various aspects of the applied references together to reject Applicant's claims. Furthermore, Applicant contends that there is no reason to modify the teachings of the Drews reference with those of either of the IEEE or Kozen references. For at least these reasons, Applicant submits that the applied references lack any teaching to suggest the limitations recited, for example, by Applicant's currently amended claim 1.

Applicant submits that the Kozen teachings cannot be readily applied to boot code, contrary to the Examiner's suggestion otherwise. The Kozen approach generates the certificate on code resulting from compilation of a **high level program.**[1] A high-level program may be understood to be a program coded in accordance with a high-level programming language, such as C++, Java, Visual Basic, etc, that exhibits strong type safety. In fact, Kozen explicitly limits application of the Kozen techniques to strongly type safe, high-level programming languages when stating that "[o]ur approach works *only* for type-safe languages."[2] Yet, Applicant submits that boot code is well-known as a low-level program, most of which are programmed according to programming languages lacking any sort of type safety. Accordingly, the Examiner cannot cavalierly state that Kozen applies to boot code, when in fact substantial, non-obvious, steps must be taken to enable active verification of boot code.

In order to facilitate the Examiner's understanding of these substantial, non-obvious steps, Applicant directs the Examiner to Applicant's FIG. 5, as well as, the accompanying paragraphs [0058]–[0070] of Applicant's specification. Looking to Applicant's FIG. 5, it is apparent that a high-level program is first compiled to render bytecode and a certificate. The bytecode and certificate ***are further processed by a translator, which then generates low-level code, while also adding information to the certificate***. Finally, a tokenizer processes the low-level code to generate the boot code, which is then loaded into the memory module of a peripheral device along with the certificate. Kozen, at best, provides for generating a certificate

[1] Page 4, first paragraph of section 2.1 "Block Structure."
[2] Page 15, first full paragraph under the heading "Strong typing vs. runtime checks" (Emphasis added).

based on the high-level program, but provides no teaching to suggest the further steps performed by Applicant's translator or tokenizer, which enable a certificate to define proofs that verify the safety and security of boot code for initializing a peripheral, as required by Applicant's currently amended claim 1, even though the boot code is not a high-level, type-safe language, as required by Kozen.

In rejecting Applicant's claim 1, the Examiner, however, states that pages 2–3 of Kozen are directed to "a description of the operation of the boot code being verified." Applicant disagrees, noting, as described above, that Kozen limits its application to strong-typed or type-safe, high-level programming languages. Again, Applicant can find no reference in Kozen that the certificate may describe operation of boot code, let alone operation of boot code for initializing a peripheral, as required by Applicant's currently amended claim 1. Moreover, Kozen specifically teaches away from generating a certificate to evaluate the operation of boot code in initializing a peripheral by stating that the Kozen approach applies *only* to strong-type languages. The Examiner has therefore improperly construed Kozen to read on Applicant's currently amended claim 1.

The IEEE reference does not cure this deficiency of Kozen. In fact, the IEEE reference provides no teaching concerning certificates nor does the IEEE reference even mention performing security checks in accordance with a certificate. Yet, the Examiner states, in rejecting Applicant's claim 1, that the IEEE reference teaches to downloading boot code from a peripheral device and executing the boot code in response to a check of the boot code. Applicant can find no reference to executing the boot code in response to a security check of the boot code in the IEEE reference. Similar to Kozen, Applicant submits that the Examiner has also improperly construed the IEEE reference to read on Applicant's currently amended claims.

Despite the sufficiency of these arguments to overcome the Examiner's rejection of Applicant's claim 1 under 35 U.S.C. 103(a), the Applicant has amended claim 1 to clarify that the boot code is generated from a first programming language and that the certificate includes an annotation defining a proof of security and safety for both (i) one or more blocks of code generated from a second programming language different from the first programming language and (ii) one or more corresponding blocks of the boot code resulting from translation of the one or more blocks of the code of the second programming language into the first programming

language. For example, Applicant submits that the boot code may be generated from programs coded in accordance with a low-level, non-object oriented Forth programming language as described in the present application, while the certificate includes annotation information generated from a high-level, type-safe, object-oriented Java programming language.

Applicant also notes that this amendment presents a relationship between the first and second programming languages, as currently amended claim 1, for example requires that the one or more blocks of the first programming language result from the translation of the one or more blocks of code of the second programming language. To illustrate, Java code is compiled into one or more blocks of code, for example, and produces compilation information, which is incorporated into the certificate as the annotation information. This annotation information may be used to independently verify the one or more blocks of code. The code is then translated by the translator and tokenized into the one or more blocks of boot code. This annotation information produced during compilation of the Java code is then used by Applicant's invention to verify the one or more blocks of the boot code of the first programming language. As a reulst, the annotation information can be used to independently verify ***both*** the one or more blocks of the code and the one or more blocks of the boot code. The combination of references cited by the Examiner neither teach nor suggest either of the two approaches.

The Kozen approach, as described above, applies only to high-level programming languages, not low level programming languages, such as Forth. The IEEE reference teaches to device drivers programmed in accordance with the Forth programming language, but fails to provide any teachings directed to performing a security check. Thus, these applied references lack any teaching to suggest each and every limitation set forth in Applicant's currently amended claim 1. However, as a result of Kozen's teaching away from application of the Kozen approach to low-level languages, it is impermissible to combine the teaching of Kozen with that of the IEEE reference. That is, the Kozen reference requires a high-level programming language that is type-safe as the Kozen technique takes advantage of these features. It provides no teaching as to how the techniques can be applied to a lower-level language that does not have such features. As noted above, Kozen explicitly states that "[o]ur approach works *only* for type-safe languages."[3]

[3] Page 15, first full paragraph under the heading "Strong typing vs. runtime checks" (Emphasis added).

The applied references therefore cannot be combined, nor individually construed, to reach Applicant's invention as set forth in currently amended claim 1.

From these improper constructions, Applicant presumes that the Examiner is improperly utilizing hindsight to combine, in a piecemeal fashion, the applied references so as to reach Applicant's claims. Once again, Kozen emphasizes that the Kozen approach is strictly limited to high-level programs having strong type safety, not low level languages. Kozen provides no teaching that enables the techniques to be adapted to address low-level code. In fact, Kozen specifically teaches that the Kozen approach will only work when applied to languages exhibiting strong type safety. The IEEE reference is concerned with addressing problems associated with loading an operating system having a plurality of user-installed input / output (I/O) devices, not with providing security checks to verify that a device driver operates as it is intended to operate.[4] Thus, neither the IEEE reference nor Kozen teach much less suggest a motivation that would lead one with skill in the art to combine these two references to reach Applicant's claimed invention. Quite the opposite, Kozen teaches away from such a combination.

Applicant contends that the invention set forth in Applicant's claim 1 overcomes the deficiencies of Kozen in that Applicant's invention enables application of active software verification to boot code that controls operation of a peripheral device, such as that described by the IEEE reference. That is, Applicant's invention as set forth in claim 1, for example, enables application of the Kozen approach to boot code *despite the teachings of Kozen to the contrary*. For the Examiner to contend otherwise suggests that the Examiner is working backwards from Applicant's invention, i.e., using hindsight, to form the rejection instead of working from the prior art forwards to reach Applicant's invention.

While Applicant acknowledges that no express, written motivation to combine must appear in the prior art references before a finding of obviousness per MPEP 2145, the motivation however may not be found when the references teach away from the claims. MPEP 2143.01 provides that the proposed modification cannot render the prior art unsatisfactory for its intended purpose or change the principle of operation of a reference. Here, suggesting that the Kozen

[4] IEEE, page 1, section 1.2 titled "Firmware problems."

approach can be modified to apply to low-level programs requires a significant modification that changes the principle of operation of the Kozen reference. This is impermissible.

Applicant further contends that there is no reason to modify the teachings of the Drews reference with those of either of the IEEE or Kozen references. Drews teaches to a security system that works in conjunction with existing systems that attempt to eliminate a component of computers most likely to fail, e.g., a memory.[5] The Drews security system requires that these existing systems provide a boot image as well as a manifest during a pre-boot operational state of a remote computer.[6] The Drews security system verifies that the boot image "has not been altered" based on hash values and digital signatures included within the manifest prior to loading the boot image.[7] Drews therefore is directed to a security system for verifying, during a pre-boot operational state, whether the boot image has been altered using hash codes and digital signatures of the manifest. Drews, however, fails to suggest or even imply that a security threat exists with respect to peripheral devices, leaving Applicant to wonder why one with skill in the art would modify Drews in the manner suggested by the Examiner.

In rejecting claim 1, the Examiner fails to provide any motivation concerning why one with skill in the art would modify Drews to perform the teaching of either Kozen or the IEEE reference. The Examiner merely suggests that one with skill in the art would combine the IEEE method with that of Kozen, leaving Applicant to presume some motivation for applying this combined IEEE / Kozen teaching to Drews. Applicant notes again that no motivation exists to combine the teachings of the IEEE reference with that of Kozen, let alone combine the resulting IEEE / Kozen teachings with those of Drews. Drews is directed to downloading and verifying boot images not boot code to control a peripheral device, as required by Applicant's claim 1. Moreover, device drivers are stored within a memory of the peripheral and not within a server so Drews only applies, at best, tenuously to Applicant's invention. Applicant therefore submits that there is little if any reason to combine the teachings of the IEEE and Kozen references to that of Drews.

---

[5] Column 1, lines 45-67.
[6] Column 1, lines 61-67.
[7] Column 2, lines 32-35.

Drews also fails to cure the deficiencies described above with respect to Kozen. The Drews reference does not suggest that the certificate may apply to low-level programs, much less that a certificate describing the operation of boot code may apply to low-level programs. In fact, the Examiner, in rejecting Applicant's independent claim 1 relies on Kozen to suggest a certificate that describes the operation of the boot code being verified. Thus, even if Drews were to teach to application of the certificate to low-level programs, Kozen specifically teaches away from application of a certificate generated in accordance with the Kozen approach to verify low-level programs. As a result, the applied references fail to teach or suggest each and every limitation recited by Applicant's currently amended claim 1.

Drews in view of Kozen and the IEEE reference further lack any teaching to suggest each of Applicant's currently amended independent claims 18 and 47.

With respect to claim 18, the applied references, for at least some of the reasons described above, lack any teaching to suggest a device comprising an interface to retrieve boot code and a certificate from a peripheral device upon power-up of the device, wherein the boot code is generated from a first programming language, and wherein the certificate includes annotation information defining independently verifiable proofs of security and safety of one or more blocks of code generated from a second programming language different from the first programming language.

Further, the applied references lack, for at least some of the reasons described above, any teaching to suggest the device also comprising a control unit to verify security of the boot code associated with the peripheral device by performing a security check on one or more blocks of the boot code in accordance with the annotation information of the certificate, the control unit configured to execute the boot code to (i) initialize the peripheral device based on a result of the security check and (ii) provide, subsequent to the initialization, an interface by which the control unit controls operation of the peripheral device, as required by Applicant's currently amended claim 18.

With respect to claim 47, the applied references lack, for at least some of the reasons described above, any teaching to suggest a computer-readable medium comprising instructions for causing a programmable processor to retrieve boot code from a peripheral device, wherein the

boot code is generated from a first programming language, and store the boot code on a computer coupled to the peripheral device.

The applied references further, for at least some of the reasons described above, lack any teaching to suggest the computer-readable medium also comprising instructions for causing the programmable processor to verify security of the boot code associated with the peripheral device by performing a security check on the boot code in accordance with a certificate that describes operation of the boot code, wherein the certificate includes an annotation defining a proof of security and safety for both (i) one or more blocks of code generated from a second programming language different from the first programming language and (ii) one or more corresponding blocks of the boot code resulting from translation of the one or more blocks of the code of the second programming language into the first programming language, and execute the boot code based on a result of the security check to (i) initialize the peripheral device and (ii) provide, subsequent to the initialization, an interface by which the programmable-processor controls operation of the peripheral device, as required by Applicant's currently amended claim 47.

Applicant's claims 2–5, 7–8, 19–22, 24, 25, 40, 48–51, and 53–54 also benefit from the arguments made above with respect to independent claims 1, 18 and 47 by virtue of depending respectively from these independent claims 1, 18, 47.

In the Office Action, the Examiner rejected claims 6, 23 and 52 under 35 U.S.C. 103(a) as being unpatentable over Drews (US 6,463,535) in view of IEEE Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practice (hereafter "IEEE") and Kozen "Efficient Code Certification," in further view of Rudoff et al. (US 6,364,378). Applicant respectfully traverses the rejection to the extent such rejections may be considered applicable to the claims as amended. The applied references fail to disclose or suggest the inventions defined by Applicant's claims, and provide no teaching that would have suggested the desirability of modification to arrive at the claimed invention.

Applicant submits that Rudoff fails to cure the deficiencies described above with respect to Drews in view of IEEE and Kozen. The Examiner, in rejecting claims 6, 23 and 52, relies on Rudoff only for its teachings concerning the IEEE-1275 Open Firmware standard. Applicant notes the IEEE reference cited by the Examiner corresponds to the Open Firmware standard. Applicant directs the Examiner to the top left corner on page 2 of the IEEE reference, which

expressly identifies the document as "IEEE Std 1275-1994." Applicant submits that the Rudoff reference is redundant, but still notes that Rudoff fails to cure the deficiencies described above with respect to Kozen, IEEE and Drews.

In particular, Rudoff fails to provide any teaching to overcome the limitation presented by Kozen that the Kozen approach applies only to type-strong or type-safe, high-level programming languages. Rudoff furthermore lacks any teaching to suggest that the boot code may be generated from a first programming language, while the certificate includes information generated from a second programming language, as set forth by Applicant's currently amended claims 1, 18 and 47. Rudoff also fails to provide any motivation concerning why one with skill in the art would combine Kozen and the IEEE reference with that of Drews. Rudoff further yet fails to provide any teaching to suggest verifying boot code prior to loading such boot code. Rudoff instead focuses on "developing programs to detect devices on an I/O bus and build a device tree that is compliant with IEEE-1275 standards and determines the device configuration."[8]

In the Office Action, the Examiner rejected claims 35-41 under 35 U.S.C. 103(a) as being unpatentable over Drews (US 6,463,535), IEEE Standard for Boot (Initialization Configuration) Firmware: Core Requirements and Practice (hereafter IEEE) in view of Kozen "Efficient Code Certification" and Ong (US 2004/0177258). Applicant respectfully traverses the rejection to the extent such rejections may be considered applicable to the claims as amended. The applied references fail to disclose or suggest the inventions defined by Applicant's claims, and provide no teaching that would have suggested the desirability of modification to arrive at the claimed invention.

Applicant submits that at least some of the arguments made above with respect to claim 1 apply to claim 35, as claim 35 recites and has been amended to recite many if not all of the same limitations as that of claim 1.

For example, the applied references lack any teaching to suggest a system comprising a peripheral device having a memory module, wherein the memory module stores a boot code and a certificate, wherein the boot code is generated from a first programming language, and wherein

[8] Column 3, lines 23-28.

the certificate includes an annotation defining a proof of security and safety for both (i) one or more blocks of code generated from a second programming language different from the first programming language and (ii) one or more corresponding blocks of the boot code, as required by Applicant's currently amended claim 35.

The applied references further lack any teaching to suggest the system comprising a computer having an interface to retrieve the boot code and the certificate from the peripheral device, a second memory module and a control unit, wherein the control unit uses the interface to retrieve the boot code and the certificate from the peripheral device and executes a verification module that verifies security of the boot code by performing a security check on the boot code to independently verify the proof represented by the annotation information of the certificate, and wherein the control unit further executes the boot code based on a result of the security check to (i) initialize the peripheral device and (ii) provide, subsequent to the initialization, an interface by which the control unit controls operation of the peripheral device, also as required by Applicant's currently amended claim 35.

As a result, Applicant, once again, notes that Ong fails to overcome the deficiencies described above with respect to the rejection of Applicant's claim 1. Ong describes a method and apparatus for user authentication, not verification of boot code that is used to control a peripheral device, as required by Applicant's currently amended claim 1.[9] The Ong certificate, the Examiner references in rejecting claims 34-41, refer to digital certificates required to authenticate the identity of a user not those that describe operation of boot code.[10]

As a result being directed to user authentication, the Ong reference provides no teaching to overcome the severe deficiency of Kozen that the Kozen approach can only be applied to strong-typed or type-safe, high-level programming languages. Ong is simply unconcerned with verifying boot code to control a peripheral device and instead focuses on securing user information to prevent hacking. Moreover, Ong provides no motivation to suggest such application to Drews. In fact, one with skill in the art would be confused to combine Drews, which pertains to verifying boot images as a whole, with the IEEE reference, which pertains to defining an open firmware standard, with Kozen, which pertains to verifying *only* strong-typed,

[9] Ong, paragraph [0014].
[10] Paragraph [0025]

high-level programming languages, with Ong, which pertains to "physicalization of user credentials at a hardware device."[11] Ong therefore lacks any teaching to overcome the deficiencies described above.

Applicant again reiterates that the rejection appears as a piecemeal combination of unrelated aspects of the various applied references that fails to consider the full teachings of each reference. That is, Kozen applies to verification of high-level programming languages, while the IEEE reference provides a standard for low-level boot code, i.e., firmware, unconcerned with security. The Drews reference applies to verifying boot images prior to loading the boot image, but such techniques rely on antiquated checksum and hash checks, which Kozen clearly replaces, but only for high-level programming languages. The Rudoff reference provides little if any additional teachings over the IEEE reference and the Ong reference applies to ways for securing user authentication information. Finding any sort of common ground or teaching on which to combine these references appears difficult at best, and the motivations proffered by the Examiner appear to recite simple benefits that each reference provides, but little in the way of actual motivations or problems that might cause one with skill in the art to combine these references. Thus, Applicant again submits that the Examiner has improperly used hindsight to combine these references.

Applicant further submits that claims 36-41 benefit equally from the arguments made above with respect to claim 35 by virtue of depending from independent claim 35.

For at least these reasons, the Examiner has failed to establish a prima facie case for non-patentability of Applicant's claims 1–8, 18–25, 35–41, 47–54 under 35 U.S.C. 103(a). Withdrawal of this rejection is requested.

New Claims:

Applicant has added claims 65–69 to the pending application. The applied references fail to disclose or suggest the inventions defined by Applicant's new claims, and provide no teaching that would have suggested the desirability of modification to arrive at the claimed inventions. As one example, the references fail to disclose or suggest that the first programming language

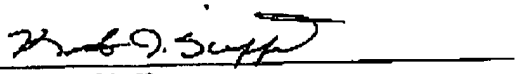
[11] Ong, paragraph [0014].

comprises a non-object oriented programming language, and the second programming language comprises an object oriented programming language, as recited by each of claims 65–68. That is, the certificate is generated from an object oriented programming language, such as Java, but the boot code to be verified is generated from a non-object oriented programming language, such as Forth. None of the above applied references teach or suggest this additional limitation. No new matter has been added by the new claims.

CONCLUSION

All claims in this application are in condition for allowance. Applicant respectfully requests reconsideration and prompt allowance of all pending claims. Please charge any additional fees or credit any overpayment to deposit account number 50-1778. The Examiner is invited to telephone the below-signed attorney to discuss this application.

Date: June 11, 2008

By: 

SHUMAKER & SIEFFERT, P.A.
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

Name: Kent J. Sieffert
Reg. No.: 41,312